



Attorney Docket No. 042390.P6518
Express Mail No: EM522828818US

MARKED-UP ORIGINAL SPECIFICATION
UNITED STATES PATENT APPLICATION

FOR

RETRIEVING I/O PROCESSOR
PERFORMANCE MONITOR DATA

RECEIVED
APR -6 2000
TECH CENTER 2700

Inventor(s) :

Susan C. KROMENAKER
Mark L. BROWN
Linda M. ROBERTS
William C. ARTHUR, JR.

RECEIVED
APR 06 2000
TECHNOLOGY CENTER 2800

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025-1026
(310) 207-3800

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to computer system architecture. More specifically, the present invention relates to
5 retrieving performance monitor data from an I/O processor.

2. Background Information

Electronic products may be thought of as those products that involve the controlled conduction of electrons or other charge carriers, especially through microprocessors. Examples of
10 electronic products may include radios, computers, work stations, and servers as well as those involved in high-end networking and storage technology. Just about all electronic products employ one or more microprocessors disposed within a chip located on a printed circuit board. These microprocessors engage a computer
15 operating system as well as applications. The main central processing unit within the chip ~~includes~~ may include a host system. It ~~is~~ may be this host system that runs the computer operating system and the applications.

One type of processor within the host system ~~is~~ may be a host
20 processor having a host memory. Another type of processor that may be within the host system ~~is~~ may be an ~~input~~ input-output (I/O) processor. The I/O processor or I/O Platform (IOP) ~~is~~ may be a component of the host system that connects the host system memory to an I/O device to process I/O transactions. The I/O
25 device may be a part of or external to the host system through at least one of a ~~primary~~ first bus and a ~~secondary~~ second bus.

Examples of I/O devices include storage devices such as a small computer systems interface (SCSI) controller for a disk and networking devices such as an Ethernet controller.

One main function of a host system ~~is~~ may be to transmit data
5 between the host memory and an I/O device via the I/O processor.
Transmitted data ~~includes~~ may include application data, local area network (LAN) packets, and contents stored on a disk. To accomplish data transmission, data handling and processing units such as a core processor and a local memory ~~are~~ may be included
10 within the I/O processor. The core processor and the local memory ~~are~~ may be coupled to each other through an internal bus and to a messaging unit and a direct memory access unit through that same internal bus. Ideally, the system performs within established parameters.

15 To measure and monitor various system parameters that contribute to the overall performance of the I/O processor, a performance monitoring unit ~~is~~ may be integrated into the I/O processor. Under current standards, the tasks of the performance monitoring unit may include compiling performance measurements on
20 ~~the three buses: the primary-first bus; the secondary-second bus;~~ and the internal bus. The measurements of the performance monitoring unit can be used to refine code for improved system level performance. However, these measurements exist in raw, binary data for which no mechanism exists that gathers and
25 compiles this raw, I/O performance monitor data into a form that readily ~~is~~ may be usable by a computer programmer or operator.

SUMMARY OF THE INVENTION

~~The present invention relates to~~ An embodiment includes
retrieving performance monitor data from an I/O processor. A
performance monitoring driver coupled to a performance monitoring
5 unit ~~is~~ may be registered as a private driver with a real time
operating system of the I/O processor. Events within the I/O
processor ~~are~~ may be selected on which to gather data. The
selected events ~~are~~ may be sent as a message request to the real
time operating system. The message request ~~is~~ may be translated
10 into the appropriate parameters based on a set of private group
parameters that ~~are~~ may be accessible by the real time operating
system. The message request ~~is~~ may be sent as a translated
request to the performance monitoring unit. The pieces of data
requested by the translated request ~~are~~ may be returned to the
15 performance monitoring driver. The pieces of data then ~~are~~ may be
sent to a location specified in the message request.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a functional block diagram of an I/O processor;

Figure 2 is a block diagram of a networking system 200;

Figure 3 illustrates a more detailed view of a host processor

5 and an I/O processor; and

Figure 4 is a flow diagram of a method of operation 500 of the invention.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 is a functional block diagram of I/O processor 10. An example of a known processor functional block diagram ~~is~~ may be illustrated and described for the Intel® i960® RM/RP Microprocessor as set out in Intel Corporation, *i960® RM/RN I/O Processor Developer's Manual*, pp. 1-1 through 1-12 (1st ed. July 1998). The description regarding **Figure 1** ~~is~~ may be based on the Intel® i960® RM/RP Microprocessor.

As show in **Figure 1**, I/O processor 10 ~~integrates~~ may integrate core processor 14 ~~into~~ as part of a Peripheral Components Interconnect (PCI) functionality so as to address the needs of intelligent input-output applications ("intelligent I/O" or "I₂O" applications). Intelligent I/O applications may be coupled to ~~primary~~ PCI bus 16 and/or ~~secondary~~ PCI bus 18. Preferably ~~both~~ Both PCI bus 16 and PCI bus 18 ~~are~~ may be industry standard, 64-bit/32-bit, high performance, low latency system buses coupled together by PCI-to-PCI bridge 20. A specification for the PCI bus ~~is~~ may be set forth in the document *PCI Local Bus Specification*, revision 2.1, October, 1994. This manual ~~is~~ may be prepared and maintained by the PCI Special Interest Group (PCI-SIG). The PCI-SIG ~~is~~ may be an organization that ~~is~~ may be open for membership to all companies in the computer industry.

Along with providing a connection path between the two independent PCI buses 16 and 18, bridge 20 ~~provides~~ may provide the ability to overcome PCI electrical loading limits by allowing certain bus transactions on one PCI bus to be forwarded to the other PCI bus. Core processor 14 ~~is~~ may be indirectly connected

to PCI-to-PCI bridge 20. Bus interface unit 24 ~~couples~~ may couple core processor 14 to internal bus 26. In turn, internal bus 26 ~~is~~ may be coupled to ~~primary~~-transfer group 28 and ~~secondary~~-transfer group 30. Internal bus 26 may be a 64-bit bus. PCI-to-PCI bridge
5 20 ~~is~~ may be coupled to ~~primary~~-transfer group 28 through ~~primary~~ PCI bus 16 and ~~is~~ may be coupled to ~~secondary~~-transfer group 30 through ~~secondary~~ PCI ~~group~~ bus 18, each path of which ~~provides~~ may provide a link to core processor 14. By communicatively connecting core processor 14 to bridge 20, core processor 14 ~~gives~~
10 may provide processor "intelligence" to bridge 20 to better address the needs of intelligent I/O applications coupled to ~~primary~~ PCI bus 16 and/or ~~secondary~~ PCI bus 18.

PCI agents 100 may be coupled to either ~~primary~~ PCI bus 16 or ~~secondary~~ PCI bus 18 so as to interact with one of the transfer
15 groups, 28 and 30. PCI agents 100 may include external PCI devices or a host processor, such as host processor 240. Within PCI agent 100 may be PCI memory having PCI address spaces.

Internal bus 26 may be coupled to local memory 38 through memory controller 40. Local memory 38 ~~includes~~ may include memory
20 systems external to I/O processor 10 that do not require external logic. Examples of local memory 38 may include Synchronous Dynamic Random Access Memory (SDRAM), Read-Only Memory (ROM), and Flash memory.

~~Primary transfer~~ Transfer group 28 ~~preferably is~~ may be
25 composed of Address Translation Unit 32, two Direct Memory Access channels 34, and messaging unit 36. ~~Secondary transfer~~ Transfer group 30 ~~preferably is~~ may be composed of Address Translation Unit

42 and One DMA channel 44.

Address Translation Unit (ATU) 32 ~~allows~~ may allow transactions between the PCI address space within PCI agent (or "agents") 100 and address space 46 within local memory 38.

5 Address translation may be controlled through programmable registers (not shown) that ~~are~~ may be accessible from both PCI agent 100 and core processor 14. ATU 42 ~~functions~~ may function similarly to ATU 32, but ~~performs on secondary~~ may work in conjunction with PCI bus 18 for PCI agents 100 coupled to
10 ~~secondary~~ PCI bus 18. Dual access to registers through ATU 32 and ATU 42 ~~allows~~ may allow flexibility in mapping the coupled address spaces.

To ~~insure~~ provide low latency and high throughput data transfers between PCI agents 100 and local memory 38, three
15 separate DMA channels ~~are~~ may be provided as shown in **Figure 1**. Two Direct Memory Access (DMA) channels 34 ~~are~~ may be included with ~~primary~~ transfer group 28 and one DMA channel 44 ~~is~~ may be included with ~~secondary~~ transfer group 30. The three DMA channels may operate as a DMA controller to support chaining and unaligned
20 data transfers. This DMA controller ~~is only~~ may be programmable through core processor 14.

The I₂O Architecture Specification describes an open architecture for developing device drivers in a system environment. Conventionally, based on the I₂O Architecture
25 Specification, messaging unit (MU) 36 provides data transfer between PCI agents 100 coupled to ~~primary~~ PCI bus 16 and core processor 14. MU 36 can be used to send and receive messages.

Moreover, MU 36 ~~interrupts~~ may interrupt PCI bus agents 100 or core processor 14 when new data arrives and passes the data as directed.

Core processor 14 ~~is~~ may be interfaced with internal bus 26 through bus interface unit 24. Local memory 38 ~~is~~ may be coupled to internal bus 26 through memory controller unit 40.

Microcontrollers 56 ~~are~~ may be interfaced with internal bus 26 through the series of Inter-Integrated Circuit (I²C) serial bus 50 and I²C bus interface unit 52. Both local memory 38 and

microcontrollers 56 ~~are~~ may be external to I/O processor 10.

Application accelerator unit 54 ~~is~~ may be also coupled to internal bus 26.

Memory controller 40 ~~allows~~ may allow direct control of local memory 38. Core processor 14 ~~operates~~ may operate out of local

memory 38 where this memory space ~~is~~ may be independent of PCI agents 100. Bus interface unit (BIU) 24 ~~forwards~~ may forward accesses to core processor 14 to internal bus 26 without

translation. Microcontrollers 56 may perform management functions for the systems of I/O processor 10. Application accelerator unit

(AAU) 54 ~~executes~~ may execute data transfers to and from local memory 38 on behalf of core processor 14 as set out for the I₂O standard.

I/O processor 10 ~~also includes~~ may further include internal arbitration unit 60 to serve as arbiter for the systems of internal bus 26, ~~secondary~~ PCI arbitration unit 62 to serve as arbiter for the ~~secondary~~ PCI bus 18, and performance monitoring unit (PMON) 64.

As noted above, performance monitoring unit (PMON) 64 may be used to compile performance measurements on the three buses:

~~primary~~ PCI bus 16, ~~secondary~~ PCI bus 18, and internal bus 26.

The compiled measurements of PMON 64 can be used for performance analysis or real-time system tuning by refining code for improved system level performance.

Within the I₂O Architecture, data ~~is~~ may be stored in parameter groups on I/O processor 10 such as in local memory 38.

Data in local memory 38 may be modified or extracted by host

processor 240 through a message sent by host processor 240 to I/O processor 10. However, the I₂O Architecture does not provide any

mechanism for gathering performance measurements compiled by PMON

64. The below embodiments of the invention extend the I₂O

Architecture to provide this capability.

Figure 2 is a block diagram of networking system 200.

Networking system 200 ~~includes~~ may include client 300 coupled to

client 400 through host system 230. Host system 230 may include

host processor 240 coupled to I/O processor 210 through host

system or PCI bus 250. Within host system 230 ~~is~~ may be I/O

device 260 interfaced with I/O processor (IOP) 210. Network lines

402 ~~are~~ may be coupled to I/O device 260.

Client 300 may be a computer that ~~includes~~ may include data

input devices such as keyboard 302 and mouse 304 and ~~includes~~ may

include visual monitor 306. Preferably, host system 230

physically ~~is~~ may be part of client 300, but may be remote from

client 300. For example, client 300 may be in one location and

host system 230 may be in another location, but connected via

communication channels 308, where channels 309 may include ~~such as~~ radio signals, cabling, or the Internet.

As one example of networking system 200, host system 230 may be connected to client 400 through network lines 402. Network
5 lines 402 may be any form of communication channel over which data from host system 230 may be transmitted to client 400. Client 400 may be composed of one computer or millions of computers.

Figure 3 illustrates a more detailed view of host processor 240 and I/O processor 210. Within host processor 240 ~~is~~ may be an
10 Operating System Specific Module (OSM) 300. Preferably, OSM 300 ~~is~~ may be the host process software for the I₂O architecture. OSM 300 ~~is~~ may be coupled to Real Time Operating System (RTOS) 302. The software of RTOS 302 ~~abstracts~~ may abstract a large portion of I/O processor 210 hardware from the rest of the software that runs
15 on I/O processor 210. By permitting common commands to enable applications within I/O processor 210, RTOS 302 ~~permits~~ may permit a programmer to develop a driver for one I/O device and get the driver running on one or many different I/O processors with minimal effort.

20 Conventionally, drivers ~~are~~ may be used to couple devices external to I/O processor 210. As shown in **Figure 3**, Small Computer System interface (SCSI) device driver 312 ~~couple~~ may couple SCSI controller 310 to RTOS 302 as an example of a storage device. ~~Networking~~ Network lines 402, as coupled to Ethernet 410,
25 ~~is~~ may be coupled to RTOS 302 through ~~networking~~ network driver 314 as an example of a ~~networking~~ networked device.

RTOS 302 ~~aids~~ may aid in message passing between the external

devices and OSM 300. By conceptually treating PMON unit 64 as a device external to I/O processor 210, the invention takes advantage of the message passing features of RTOS 302 by coupling PMON unit 64 to RTOS 302 through PMON driver 320. PMON driver 320
5 ~~is~~ may be a Device Driver Module (DDM) dedicated to performance (perf) monitoring resources and may be referred to as "perfDDM". PMON driver 320 may operate by itself or work in conjunction with either a storage device or a networking device.

The system management interface of the I₂O Architecture
10 Specification provides for developing private parameter groups. Private parameter groups may reside in the I/O processor memory. An Operating System Specific Module may be used to gather data from a driver. For example, private parameter groups of the invention ~~preferably~~ may reside in the I/O processor memory and
15 OSM 300 may be used to gather data from PMON driver 320 of **Figure 3**.

Each parameter group may include a group number, a group type, a group name, a description of the group, and ~~includes~~ may include one or more parameters belonging to the group. A
20 parameter may be identified by a field number, whether the parameter ~~is~~ may be readable or writable, the file size of the parameter, the parameter name, and a description of the parameter. In one embodiment of the invention, three private parameter groups ~~contain~~ containing a total of thirty one fields reside as software
25 in memory 38 of I/O processor 210.

Within performance monitoring Table 1, Table 2, and Table 3 below ~~are~~ may be a set of parameter groups that define an

embodiment of the invention. Table 1 ~~is~~may be the Control Group, Table 2 ~~is~~may be the Mode Control Group and Table 3 ~~is~~may be the Mode Data Group. The contents of the parameter groups may include the performance monitor data and performance monitor setup

5 information. Software running on host processor 240 ~~can~~may be able to gather or modify one or more parameters stored in these parameter groups.

Table 1 Performance Monitoring Control Group 0x8000x

Group Number		0x8000x		
Group Type		SCALAR		
Name		PERFMON_CONTROL		
Description		A table of all control parameters for hardware-based performance monitoring resources		
Field ldx	(r/w)	Field Size	Parameter Name	Description
0	r/w	5 Bytes	LockControl	<p>Bytes 0-3 are <u>may be</u> the AuthenticationKey.</p> <p>When the LockControl field is read by an application and the Locked bit is not set, the initiator Target Identification (TID) from the UtilGetParams message is <u>may be</u> saved, a unique AuthenticationKey is <u>may be</u> returned in these bytes of the field, and the Locked bit is <u>may be</u> tentatively changed to the set state. The application then has 2 seconds to lock the performance monitoring resources by sending a UtilParamsSet message with the proper TID to write to the LockControl field with the Locked bit set and the proper AuthenticationKey in Bytes 0-3. When this occurs, the resources remain locked, if they are currently unlocked. Otherwise, after the 2 second timeout period runs out, the locked bit is <u>may be</u> reset and other applications can attempt to lock the resources. During the 2 second backoff period, other applications that read the LockedControl field will detect that the resources are already locked, and the AuthenticationKey will be set to something other than the proper one.</p>

1	r/w	11 Bytes	Control	<p>Bit 0 of byte 4 is <u>may be</u> the Locked bit.</p> <p>For the Locked bit, 1 means locked and 0 means unlocked. When this bit is set along with a proper AuthenticationKey, the perfDDM saves the initiator TID, fields from the UtilParamsSet message. Every subsequent UtilParamsSet or UtilParamsGet message with an</p>
---	-----	----------	---------	--

Table 1 cont. Performance Monitoring Control Group 0x8000x

Group Number		0x8000x		
Group Type		SCALAR		
Name		PERFMON_CONTROL		
Description		A table of all control parameters for hardware-based performance monitoring resources		
Field Index	(r/w)	Field Size	Parameter Name	Description

1 cont.	r/w	11 Bytes	Control	<p>initiator TID field equal to the saved TID value causes a watchdog timer to be zeroed. After 5 min., if no UtilParamsGet or UtilParamsSet messages to the perfDDM parameter groups with the initiator TID field equal to the saved value are received, then the locked state bit is <u>may be</u> cleared. This mechanism mediates resource contention between applications and DDMS under development, while preventing resource lockout due to halted applications. Default is <u>may be</u> unlocked(0).</p> <p>NOTE: Because of the 2 second timeout condition, spurious UtilGetParams messages that read the LockControl field should be avoided, since these reads would prevent other applications from being able to lock the performance monitoring resources during the backoff period.</p> <p>Bytes 0-3 contain the AuthenticationKey. The AuthenticationKey verifys <u>verifies</u> that the configuration host application that is <u>may be</u> attempting to alter the alteration is the one that locked the resources. On writes, this key must be equal to the last authentication key issued by the perfDDM. If equal, the host application's UtilParamsSet message will be processed normally. If not equal to the last issued authentication key, then the UtilSetParams reply message indicates an error. On reads this field returns 0.</p>
------------	-----	-------------	---------	---

Table 1 cont. Performance Monitoring Control Group 0x8000x

Group Number	0x8000x			
Group Type	SCALAR			
Name	PERFMON_CONTROL			
Description	A table of all control parameters for hardware-based performance monitoring resources			
Field Index	(r/w)	Field Size	Parameter Name	Description
1 cont.	r/w	11 Bytes	Control	<p>Bit 0 of byte 4 is <u>may be</u> the Accumulate count bit.</p> <p>For the Accumulate count bit, 0 means only report deltas for last interval and 1 means accumulate counters and time intervals. Default is <u>may be</u> Accumulate counters and time intervals(1).</p> <p>Bit 1 of byte 4 is <u>may be</u> the Counting_On bit.</p> <p>The Counting_On bit gives the application a quick way to turn off counting in the performance monitor. This will be used to limit the impact of performance monitoring-related bus accesses on the counter statistics reported. When an application wants to turn counting off, it will send the smallest UtilParamsSet message possible to access only this field. When counting is turned ON, the saved counter values for each mode are <u>may be</u> zeroed out. Default is <u>may be</u> counting off.</p> <p>Note: when Counting_On is 1, all writes to any other parameter group fields or bits than Counting_ON are <u>may be</u> disallowed. An error will be returned for such accesses.</p> <p>Note: when Counting_On is 0 and all the ModelInterval fields for all modes are <u>may be</u> set to 0 (see group 0x8001, field #1 below), counting cannot be turned on, since no modes</p>

Table 1 cont. Performance Monitoring Control Group 0x8000x

Group Number Group Type Name Description		0x8000x SCALAR PERFMON_CONTROL A table of all control parameters for hardware-based performance monitoring resources		
Field Index	(r/w)	Field Size	Parameter Name	Description
1 cont.	r/w	11 Bytes	Control	<p>have been allocated counting intervals. Attempts to set Counting_On in this case will result in an 120_PARAMS_STATUS_SCALAR_ERROR being returned by the operation.</p> <p>Bit 2 of byte 4 is <u>may be</u> the GlobalSendOnCycleEnd bit. If bit 2 is set, the EventData fields of Event Acknowledge Messages for the CYCLE_END event will contain selected rows of the PERFMON_MODE_DATA table group. If this bit is reset, Event Acknowledge messages contain no EventData. Default is <u>may be</u> reset.</p> <p>Bit 3 of byte 4 is <u>may be</u> the Pause bit. The Pause bit can be used to temporarily disable counting and subsequently reenale counting without zeroing the saved counter values for each mode. Default is <u>may be</u> OFF.</p> <p>Byte 5 is <u>may be</u> the CurrAlgorithm The CurrAlgorithm is <u>may be</u> used to set the sampling algorithm. Initially, there will be two sampling algorithms supported: User-configurable simple round-robin(0), and distributed round-robin(1). Default: distributed round-robin.</p> <p>Bit 2 of byte 4 is <u>may be</u> the GlobalSendOnCycleEnd bit. If bit 2 is set, the EventData fields of Event Acknowledge Messages for the CYCLE_END event will contain selected rows of the PERFMON_MODE_DATA table group. If</p>

Table 1 cont. Performanc Monitoring Control Group 0x8000x

Group Number		0x8000x		
Group Type		SCALAR		
Name		PERFMON_CONTROL		
Description		A table of all control parameters for hardware-based performance monitoring resources		
Field Index	(r/w)	Field Size	Parameter Name	Description
1 cont.	r/w	11 Bytes	Control	<p>this bit is <u>may be</u> reset, Event Acknowledge messages contain no EventData. Default is <u>may be</u> reset.</p> <p>Bit 3 of byte 4 is <u>may be</u> the Pause bit.</p> <p>The Pause bit can be used to temporarily disable counting and subsequently reenale counting without zeroing the saved counter values for each mode. Default is <u>may be</u> OFF.</p> <p>Byte 5 is <u>may be</u> the CurrAlgorithm</p> <p>The CurrAlgorithm is <u>may be</u> used to set the sampling algorithm. Initially, there will be two sampling algorithms supported: User-configurable simple round-robin(0), and distributed round-robin(1). Default: distributed round-robin.</p> <p>Byte 6 is <u>may be</u> the SampleUnits.</p> <p>The SampleUnits specifies the units for the selected sample interval: usec(0), msec(1), sec(2), min(3). Default: msec.</p> <p>Bytes 7 through 10 are <u>may be</u> the SampleInterval; which specifies the number of SampleUnits in the selected sample interval. Default: 100, giving a default sample interval of 100 msec. If the users select as a sample interval less than the minimum sample interval, the actual sample interval is <u>may be</u> rounded up. The largest sample interval supported is <u>may be</u> 71 minutes.</p>

Table 1 cont. Performance Monitoring Control Group 0x8000x

Group Number		0x8000x		
Group Type		SCALAR		
Name		PERFMON_CONTROL		
Description		A table of all control parameters for hardware-based performance monitoring resources		
Field Index	(r/w)	Field Size	Parameter Name	Description
2	r	1 Byte	MaxMode	Maximum # of modes for the performance monitoring resources. DDMS use this to indicate the number of modes supported by the underlying hardware. Typically these are <u>may be</u> hardware modes.
3	r	1 Byte	CurrentMode	Current mode of the performance monitoring resources. This is <u>may</u> be used for debugging purposes.
4	r	1 Byte	MaxAlgorithm	Maximum # of sample algorithms. DDMS can use this to indicate the number of algorithms that can be supported. Sample algorithms are <u>may be</u> software-controlled.
5	r	1 Byte	MinSampleUnits	Specifies the units for the minimum supported sample interval: usec(0), msec(1), sec(2), min(3). This value will be determined by the resolution of the RTOS event timer.
6	r	4 Bytes	MinSampleInterval	Number of MinSampleUnits in the minimum supported sample interval. This value will be determined by the resolution of the RTOS system timer rounded to the nearest microsecond.
7	r	1 Byte	PerfHWType	Type of performance monitoring hardware available: -0 means NONE, 1 means i960 RN or RM, all other values are <u>may be</u> reserved.
11	r	1 Byte	NumCounters	Number of performance monitor counters supported by hardware.
13	r	1 Byte	SampleInterval Status	Provides an indicator of whether the user-selected sample interval is <u>may be</u> okay(0), too small(1), or too large(2), adjusted(3).

14	r	4 Bytes	AdjustedSam ple Interval	Displays the rounded up sample interval, e.g. the user-configured sample interval rounded up to the next integer multiple of the system timer resolution.
----	---	------------	--------------------------------	---

Notes for Table 1: ~~The group numbers for the performance monitoring private parameter groups preferably will be determined by the I₂O SIG assignee of the inventor. Moreover, the AuthenticationKey may be based on a global timer to guarantee uniqueness. In addition, clearing the locked stage bit after five minutes preferably will not prevent resource contention between multiple applications on the same host since messages from these all have the same initiator TID value. Originally, it was thought that the initiator context field could be used for this purpose. However, some host operating systems control this field. Applications have no control over this field.~~

Table 2 Performanc Monitoring Mode Control Group 0x8001

Group Number		0x8001		
Group Type		Table		
Name		PERFMON_MODE_CONTROL		
Description		A table of mode-specific control parameters for the performance monitoring resources.		
Field Index	(r/w)	Field Size	Parameter Name	Description
0	r	1 Byte	Mode	Performance monitoring mode to be controlled, 1-7.
1	r/w	8 Bytes	ModeControl	<p>Bytes 0-3 contain the AuthenticationKey</p> <p>The AuthenticationKey is <u>may</u> be used to verify that the host application which is attempting to alter the configuration is the one that locked the resources. On writes, this key must be equal to the last authentication key issued by the perfDDM. If equal, the host application's UtilParamsSet message will be processed normally. If not equal to the last issued authentication key, then the UtilSetParams reply message indicates an error. On reads, bytes 3-7 return 0.</p> <p>Bytes 4 - 7 of this field contain the ModeIntervals.</p> <p>The ModeIntervals represents the number of selected sample intervals dedicated to a particular mode when using the simple round-robin; writes to this sub-field cause all modes counter values to be zeroed, if not already done. A flag, counts_zeroed, in perfDDM will track this; it will <u>may</u> be set when the first Mode Time is zeroed, and reset when counting is turned on. This will reduce the impact on the overall system that would occur if the Mode Times for more than one mode were altered. This defaults to 10 times the sampling interval.</p>

~~Note for Table 2: No rowDelete or rowAdd operations need be supported by perfDDM since these fields are present for all modes.~~

Table 3 Performance Monitoring Mode Data Group 0x8002

Group Number Group Type Name Description		0x8002 Table PERFMON_MODE_DATA A table of data parameters for all counters in each mode. Each row pertains to an individual mode.		
Field Index	(r/w)	Field Size	Parameter Name	Description
0	r	1 Byte	Mode	Performance monitoring mode for the data, 1 - 7.
1	r	8 Bytes	CurrTime	Current accumulated value of GTSR register plus rollover bits. This value accumulates over multiple sample intervals until the end of the timeslice <u>time slice</u> may be reached. This is <u>may be</u> updated whenever an interval ends, if the host application disables counting in the middle of an interval, or if the host application requests the data in the middle of an interval. Main use of this is <u>may be</u> to determine how stale or current the data is <u>may be</u> when very long interval times are <u>may be</u> being used. Units are <u>may be</u> the period of the GTSR clock.
2	r	8 Bytes	SigmaTime	Accumulated time for this mode at end of last completed interval. Units are <u>may be</u> the period of the GTSR clock.
3	r	8 Bytes	EndingTime	Value of GTSR plus rollover bits at the end of the last completed interval. This is <u>may be</u> compared to CurrTime to determine the relative currency of the counter data.
4	r	8 Bytes	Counter01	PEC01 counter value at end of last completed interval.
5	r	8 Bytes	Counter02	PEC02 counter value at end of last completed interval.
6	r	8 Bytes	Counter03	PEC03 counter value at end of last completed interval.
7	r	8 Bytes	Counter04	PEC04 counter value at end of last completed interval.
8	r	8 Bytes	Counter05	PEC05 counter value at end of last completed interval.
9	r	8 Bytes	Counter06	PEC06 counter value at end of last completed interval.

10	r	8 Bytes	Counter07	PEC07 counter value at end of last completed interval.
----	---	------------	-----------	--

Table 3 cont. Performance Monitoring Mode Data Group 0x8002

Group Number		0x8002		
Group Type		Table		
Name		PERFMON_MODE_DATA		
Description		A table of data parameters for all counters in each mode. Each row pertains to an individual mode.		
Field1 dx	(r/w)	Field Size	Parameter Name	Description
11	r	8 Bytes	Counter08	PEC08 counter value at end of last completed interval.
12	r	8 Bytes	Counter09	PEC09 counter value at end of last completed interval.
13	r	8 Bytes	Counter10	PEC10 counter value at end of last completed interval.
14	r	8 Bytes	Counter11	PEC11 counter value at end of last completed interval.
15	r	8 Bytes	Counter12	PEC12 counter value at end of last completed interval.
16	r	8 Bytes	Counter13	PEC13 counter value at end of last completed interval.
17	r	8 Bytes	Counter14	PEC14 counter value at end of last completed interval.

Notes for Table 3: ~~SigmaTime and all counter values are initialized to zero. Also, if delta counting is selected, when the performance monitor mode is first entered, Sigma time and all the counter values are zeroed. In cumulative counting, the Sigma time and all of the counter values are not zeroed when the mode is first entered, and when the mode is exited the values in the GTSR and all of the counters are added to the values in this parameter group.~~

Figure 4 is a flow diagram of a method of operation 500 of the invention. Method 500 may be implemented in a computer readable storage medium containing executable computer program instructions which when executed cause the networking system to perform method 500. Also, method 500 may be implemented in a distributed readable storage medium containing executable computer program instructions which when executed cause an I/O processor to perform method 500.

At initialization 502 of method 500, the software in local memory 38 of I/O processor 210 ~~initializes~~ may initialize. This ~~creates~~ may create a performance monitoring (PMON) storage table in local memory 38. The PMON storage table ~~can~~ may be able to store and keep separate the various pieces of information retrieved from PMON unit 64 and placed in that table.

Additionally, initializing the software in local memory 38 also ~~causes~~ may cause PMON driver 320 to register as a private driver with RTOS 302 of I/O processor 210 since PMON driver 320 ~~is not~~ may not be a networking type driver or a storage type driver.

In operation, a user of client 300 or client 400 of **Figure 2** ~~initiates~~ may initiate a performance monitor application at step 506 that ~~generates~~ may generate a selection screen at visual monitor 306. The selection screen ~~allows~~ may allow the user to select those events on ~~primary~~ PCI bus 16, ~~secondary~~ PCI bus 18, or internal bus 26 for which the user desires to compile data. The user may also specify the time periods that the user desires to see the compiled data at visual monitor 306. In regards to the events on ~~primary~~ PCI bus 16, ~~secondary~~ PCI bus 18, or internal

bus 26, the I₂O Architecture divides monitorable events into occurrence events and duration events. Occurrence events ~~is~~ may be counted each time the event occurs. For a duration event, a counter ~~counts~~ may count the number of clocks during which a particular condition or set of conditions ~~is~~ may be true. A total of ninety-eight events may be monitored, subject to the availability of event counters.

At step 510, the user ~~selects~~ may select at visual monitor 306 those events on ~~primary~~ PCI bus 16, ~~secondary~~ PCI bus 18, or internal bus 26 for which the user desires to compile data by entering their selection in the selection screen. After selecting those events the user desired monitored, the information entered into the selection screen at visual monitor 306 ~~is~~ may be sent at step 514 to OSM 300 of host processor 240 as a message request. This message request may specify ~~that specifies~~ specific pieces of data from PMON unit 64. OSM 300 ~~relays~~ may relay this message request to RTOS 302 at step 518 without an understanding of whether the message request is for a networking type driver, a storage type driver, or a private driver. However, RTOS 302 ~~does~~ may recognize this message request as a request for the PMON driver 320, previously registered as a private driver with RTOS 302.

Using the fields of the three private parameter groups residing in local memory 38 of I/O processor 210, RTOS 302 ~~translates~~ may translate the message request at step 522 into the appropriate parameter of the private group parameters of the invention. Parameters of the private group parameters ~~are~~ may be

set out in Table 1, Table 2, and Table 3 above.

At step 526, RTOS 302 ~~sends~~ may send this translated request to PMON driver 320. In response to receiving the translated request, PMON driver 320 ~~queries~~ may query PMON unit 64 at step 530 for the specific pieces of data requested by the user. The query to PMON unit 64 ~~results~~ may result in the requested pieces of data being sent to PMON driver 320 at step 534. PMON driver 320, in turn, ~~sends~~ may send this data to the PMON storage table in local memory 38 at step 538 where the data ~~will~~ may be compiled and stored for dispatch to the user at the time periods specified by the user at visual monitor 306 in the selection screen. The data ~~preferably is~~ may be saved in such a way that another application ~~could~~ may be written that would take this data and present it to client 300, for example, in a meaningful fashion.

At the time periods specified by the user in the selection screen, I/O processor 210 ~~sends~~ may send the compiled performance monitoring data back to the user through RTOS 302 and OSM 300 at step 542. The compiled performance monitoring data may also be directed to other locations within networking system 200 for purposes such as to cause an effect. For example, the data may be sent to an interpreting device that determines whether the server of host system 230 is too hot based on the compiled performance monitoring data. If so, the interpreting device may generate a message that causes an internal fan to turn on.

The exemplary embodiments described herein are provided merely to illustrate the principles of the invention and should not be construed as limiting the scope of the subject matter of

the terms of the claimed invention. The principles of the invention may be applied toward a wide range of systems to achieve the advantages described herein and to achieve other advantages or to satisfy other objectives, as well.